الوراثة الجزئية partial inheritance

المفهوم، المميزات، العيوب

م. وائل حسن- أبو إياس

المفهوم و المميزات

حينما كنتُ أقوم بتصميم لغة البرمجة إبداع في الفترة الأولي من حياة مشروع "البرمجة بإبداع"[1] (أي منذ حوالي العامين و النصف): كان تصميمها المبدئي يختلف للغاية عن التصميم النهائي الذي استقرّتْ عليه حالياً، و الحق أن إبداع شهدت علي الأقل شكلين مختلفين تمام الاختلاف عن بعضهما البعض، لدرجة أنه يمكن اعتبار كل واحدٍ منهما لغةً مختلفةً قائمةً بذاتها ! ؛ فقد كنتُ في كل فترةٍ أعيد تقييم آرائي العلمية بما يتناسب مع ما حصّلتُه من معرفةٍ جديدة، و من ثم أقوم بإعادة تقييم التصميم السابق بما يتفق مع ما يستجد عندي من آراءٍ صرتُ أقتنع بها بعد أن كنتُ أري ما يُخالفها.

و أثناء تلك الفترات التي كانت تتغير فيها قَناعاتي وجدتُ أن هناك بعض المُكوِّنات (أعني: قواعد و تعبيرات) كنتُ مقتنعاً بأهميتها في الأشكال القديمة من إبداع، و لكن بعد التفكير الجيد فيها وجدتُ أنه من الأفضل ألا يتم ضمها إلي اللغة في نسختها النهائية.
و كان من بين تلك المُكوِّنات التي تم التخلي عنها ما أطلقتُ عليه اسم "الوراثة الجزئية partial inheritance" أو "الوراثة الناقصة incomplete inheritance"، و هو نوعٌ من أنواع الوراثة في البرمجة الكائنية object oriented يبدو غريباً للغاية عند شرحه (كما سترون فيما يلي من توضيح بمشيئة الله تعالي).

ففي البداية حينما فكرتُ فى أمر التوارث و فائدته في البرمجة[2] (من حيث تحسين تنظيم الأكواد، و إعادة الاستخدام reusability و غيرهن من الفوائد) خَطَر على بالى موقفٌ ليس من الجيد استخدام الوراثة الكاملة (أي الشكل التقليدي من الوراثة) فيه، و بدا لي أنه من الأفضل استخدام الوراثة الجزئية للتبسيط على المبرمج قدر الإمكان.

1 الموقع الرسمي للمشروع http://ebda3lang.blogspot.com/

2 أرجو مُراجعة ما كتبتُه عن الوراثة في كتابي "رسالة البرمجة بإبداع"، في باب "روح الإبداع"، فصل "الوراثة" صفحة 186 من الإصدارة 1.2 .

و فيما يلى أعرض هذا الموقف:

فنفترض أن لدينا فى برنامجنا ستة أصنافٍ classes هى: صنف 1 ، صنف 2، صنف 3، صنف 4، صنف 5، صنف 6، و يحتوى كل صنفٍ منهن على سبع إجراءات[3]، و بعض هذه الإجراءات مُشترَكٌ بين هذه الأصناف[4]، و هي الإجراءات: إجراء 1 ، إجراء 2، إجراء 3، إجراء 4، إجراء 5.

مع العلم أن الأصناف تختلف في عدد الإجراءات التي تحتويها من مجموعة الإجراءات المُشترَكة السابقة؛

فالصنفان صنف 1 و صنف 2 يحتويان على الإجراءات: إجراء 1 ، إجراء 2، إجراء 3، إجراء 4، إجراء 5، و الأصناف صنف 3 و صنف 4 و صنف 5 تحتوى على الإجراءات: إجراء 1 ، إجراء 2، إجراء 3، إجراء 4، في حين يحتوى الصنف صنف 6 على الإجراءات: إجراء 1 ، إجراء 2، إجراء 3 فقط.

أي أن الأصناف ستكون كما يلي (مع ملاحظة أن الرمز **غ.م** يعنى **غير مُشترك**):

<u>صنف1</u>	<u>صنف2</u>	<u>صنف3</u>
* إجراء1	* إجراء1	* إجراء1
* إجراء2	* إجراء2	* إجراء2
* إجراء3	* إجراء3	* إجراء3
* إجراء4	* إجراء4	* إجراء4
* إجراء5	* إجراء5	* إجراء3-1 غ.م
* إجراء1-1 غ.م	* إجراء2-1 غ.م	* إجراء3-2 غ.م
* إجراء1-2 غ.م	* إجراء2-2 غ.م	* إجراء3-3 غ.م

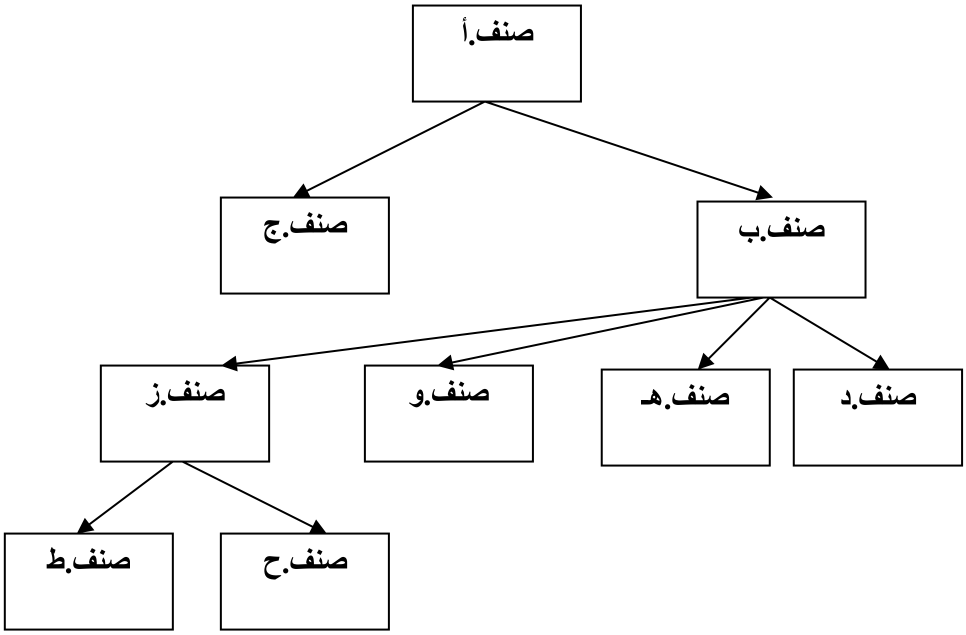
<u>صنف4</u>	<u>صنف5</u>	<u>صنف6</u>
* إجراء1	* إجراء1	* إجراء1
* إجراء2	* إجراء2	* إجراء2
* إجراء3	* إجراء3	* إجراء3
* إجراء4	* إجراء4	* إجراء6-1 غ.م
* إجراء4-1 غ.م	* إجراء5-1 غ.م	* إجراء6-2 غ.م
* إجراء4-2 غ.م	* إجراء5-2 غ.م	* إجراء6-3 غ.م
* إجراء4-3 غ.م	* إجراء5-3 غ.م	* إجراء6-4 غ.م

---

3 الإجراءات في لغة إبداع تُكافيء الدوال functions في اللغات الأخري مع بعض الاختلافات في الصياغات و المفاهيم.

4 أي أن كل إجراءٍ من تلك الإجراءات يُوجَد في كل تلك الأصناف، و ليس لصفة مشترك هنا نفس المعني الذي لها في قواعد لغة إبداع.

و لو أردنا وضع تصميمٍ لهيكل شجرة الوراثة التقليدية التي ستكون بين هذه الأصناف، بحيث نستغل إمكانية التوارث بأفضل الطرق فسيكون التصميم كما يلى:



حيث في هذا النموذج تكون الأصناف في المستوى الأدنى وارثةً للأصناف التي من المستوى الأعلى و تربطها بها أسهم، و التكوينات الداخلية لتلك الأصناف ستكون كما يلي:

صنف.أ
* إجراء1
* إجراء2
* إجراء3

صنف.ب
* إجراء4

صنف.ج
* إجراء6-1 غ.م
* إجراء6-2 غ.م
* إجراء6-3 غ.م
* إجراء6-4 غ.م

صنف.د
* إجراء3-1 غ.م
* إجراء3-2 غ.م
* إجراء3-3 غ.م

صنف.هـ
* إجراء4-1 غ.م
* إجراء4-2 غ.م
* إجراء4-3 غ.م

صنف.و
* إجراء5-1 غ.م
* إجراء5-2 غ.م
* إجراء5-3 غ.م

صنف.ز
* إجراء5

صنف.ح
* إجراء1-1 غ.م
* إجراء1-2 غ.م

صنف.ط
* إجراء2-1 غ.م
* إجراء2-2 غ.م

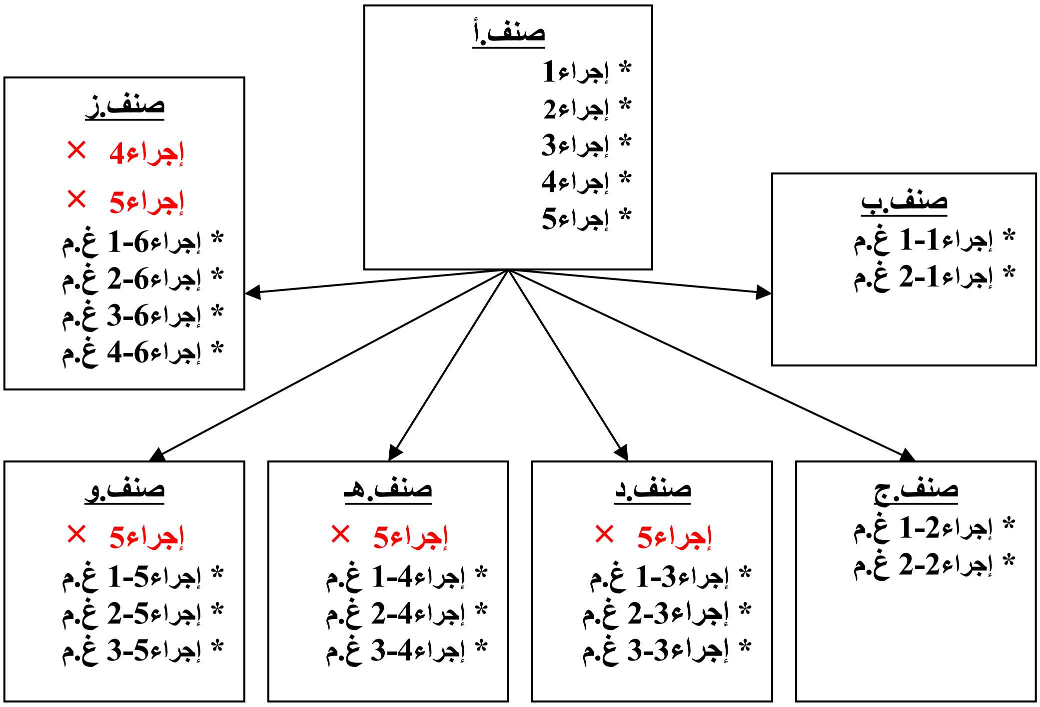
و هكذا يمكننا أن نحصل على الأصناف الرئيسة في برنامجنا: صنف 1 ، صنف 2، صنف 3، صنف 4، صنف 5، صنف 6 من الموجودة في شجرة الوراثة كما في الجدول التالي:

الصنف الرئيس	الصنف المُمثِّل له في شجرة الوراثة
صنف 1	صنف.ح
صنف 2	صنف.ط
صنف 3	صنف.د
صنف 4	صنف.هـ
صنف 5	صنف.و
صنف 6	صنف.ج

و على هذا فسيكون ثمن الوراثة الجيدة غير رخيص؛ فلكى نُمثِّل ستة أصنافٍ فقط: قمنا بكتابة تسعة أصنافٍ لتنسيق شجرة الوراثة و جعلها أفضل هيكلاً و أكثر منطقية، و هذا يعني أنه إذا ما أردنا تعديل بناء implementation إجراءٍ من الإجراءات المُشترَكة أن نتذكر في أي صنفٍ وراثي من الثلاثة أصنافٍ التنسيقية (صنف.أ، صنف.ب، صنف.ج) يُوجَد ذلك الإجراء، و إضافةً إلى ذلك فعلينا أن نُطلِق على كل صنف من الأصناف الوراثية اسماً يعبر عن وظيفته بالضبط (كما تقضى قواعد هندسة البرمجيات)، و يؤدي هذا إلي أن الأمر يصير مُرهِقاً إلي حدٍ كبيرٍ إذا ما أصبحت شبكة الوراثة أكبر و أكثر في عدد المستويات.

و عندما نفكر في كيفية استخدام الوراثة الجزئية لإيجاد تصميمٍ جيدٍ و سهلٍ للمسألة السابقة فسنجد أكثر من حل، منها:

1 - التصميم الأول:

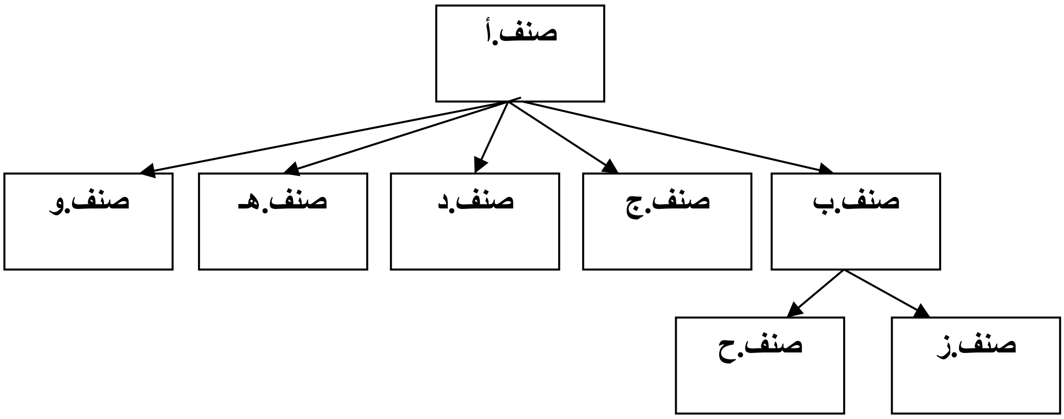


مع الأخذ في العلم أن الإجراءات التي كُتِبَت أسماؤها باللون الأحمر تم استثناؤها من الوراثة، فنحصل على الأصناف الرئيسة من أصناف شجرة الوراثة كما في الجدول التالي:

الصنف الرئيس	الصنف المُمثِّل له في شجرة الوراثة
صنف 1	صنف.ب
صنف 2	صنف.ج
صنف 3	صنف.د
صنف 4	صنف.هـ
صنف 5	صنف.و
صنف 6	صنف.ز

و بمقارنة التصميم كامل الوراثة بالتصميم السابق: نرى أن الأخير مُباشرٌ و بسيطٌ أكثر من الأول؛ حيث استخدمنا سبعة أصنافٍ فقط في البرنامج لنحصل علي الستة التي أردناها، و في نفس الوقت أَمتعنا بكل إمكانيات الوراثة و أعطانا سهولةً كبيرةً جداً في إدارة الموارد المُشترَكة بين الأصناف المتوارثة.

2- التصميم الثاني:



حيث:

صنف.أ
* إجراء1
* إجراء2
* إجراء3
* إجراء4

صنف.ب
* إجراء5

صنف.ج
* إجراء3-1 غ.م
* إجراء3-2 غ.م
* إجراء3-3 غ.م

صنف.د
* إجراء4-1 غ.م
* إجراء4-2 غ.م
* إجراء4-3 غ.م

صنف.هـ
* إجراء5-1 غ.م
* إجراء5-2 غ.م
* إجراء5-3 غ.م

صنف.و

إجراء4 ×
* إجراء6-1 غ.م
* إجراء6-2 غ.م
* إجراء6-3 غ.م
* إجراء6-4 غ.م

صنف.ز
* إجراء1-1 غ.م
* إجراء1-2 غ.م

صنف.ح
* إجراء2-1 غ.م
* إجراء2-2 غ.م

فنحصل على الأصناف الرئيسة من أصناف شجرة الوراثة كما في الجدول التالي:

الصنف الرئيس	الصنف المُمثِّل له في شجرة الوراثة
صنف 1	صنف.ز
صنف 2	صنف.ح

صنف 3	صنف.ج
صنف 4	صنف.د
صنف 5	صنف.هـ
صنف 6	صنف.و

و هو تصميمٌ يقع في المنتصف، بين التصميم الكامل الوراثة، و التصميم الأول بالوراثة الناقصة؛ فقد استخدمنا فيه ثمانية أصنافٍ لتمثيل الستة المرغوب فيهن، مع التقليل من الاستثناءات التي قمنا بها لبعض الإجراءات في عملية الوراثة إلي استثناءٍ واحدٍ فقط، و كان من نصيب الإجراء إجراء 4 في عملية وراثة صنف.و للصنف صنف.أ.

العيوب

رغم ما سبق ذِكره من أدلةٍ علي قدرة الوراثة الجزئية نظرياً علي تبسيط نموذج وراثة الأصناف بشكلٍ كبير، إلا أنه عملياً هناك عيوبٌ خطيرةٌ للغاية لهذا النوع من الوراثة يجعل من المستحيل بناءها في لغات البرمجة ذات الإمكانات الكائنية، و خاصةً تلك التي تعتمد بشكلٍ كاملٍ علي نموذج "التنويع الثابت static typing". و من تلك العيوب:

- التعارض مع أبسط مباديء البرمجة الكائنية، و أقصد مبدأ أن "الصنف الوارث يُؤدِّي جميع الوظائف التي يُؤدّيها الصنف المورث، و بنفس الخصائص المتعلقة بها[5] (و ليس من الضروري أن تكون بنفس طُرق الأداء)". أي أن الوراثة الناقصة تهدم قاعدة أن"كائنات الصنف الوارث هي كائناتٌ من الصنف الموروث" أصلاً. و هي كما قلنا من أبسط بدهيات البرمجة الكائنية و أول قواعدها التي تقوم عليها !

فعلي سبيل المثال لو تخيلنا أنه في لغة إبداع تُوجَد إمكانية الوراثة الناقصة، و أننا استخدمنا تعبير ما عدا (الذي يُستخدم في الوراثة المتعددة) للقيام بالاستثناء التام لبعض مكونات الأصناف الموروثة من عملية الوراثة: فيمكننا مثلاً الحصول علي أكوادٍ مكتوبةٍ بلغة إبداع تشبه الكود التالي:

```
صنف 1 كائن 1 = صنف 2()
كائن 1_إجراء 1()

صنف صنف 1:
    إجراء إجراء 1:
        أكتب.نص.سطر("من داخل الإجراء إجراء 1")
```

5 أقصد بالخصائص هنا أشياء مثل معدل الوصول access modifier للإجراء الموروث أو هل هو مُشترَك static أم لا، و أنواع و ترتيب المُدخلات و المُخرجات، و غيرهن من الأمور الأخري.

صنف صنف 2 يرث صنف 1 :

ما عدا:

صنف 1_إجراء 1

صنف صنف 3 يرث صنف 1 :

\ كود الصنف صنف 3 /

هذا الكود يتم تمريره في زمن التصحيح compile time بسلاسةٍ تامة، بينما في زمن التنفيذ run time ستَحدُث مشكلةٌ عند تنفيذه؛ لأن الإجراء إجراء 1 لا يُوجَد في الكائنات التي تُسنَد لها قيمةٌ من النوع صنف 2؛ لأننا استثنيناه من وراثة صنف 2 للصنف صنف 1 .

و سبب المشكلة السابقة أنه لا يمكنك معرفة هوية القيمة الحقيقية التي يحملها كائن 1 إلا في أثناء زمن التنفيذ؛ فساعتها فقط ستعلم هل تم إسناد قيمةٍ من النوع صنف 1 إليه أم من النوع صنف 2 أم النوع صنف 3، و لو كانت من النوع صنف 1 أو صنف 3 فلن تحدث هناك مشكلة في تنفيذ الكود لأن الإجراء إجراء 1 موجودٌ في كائنات كلٍ من النوعين، أما في حالة صنف 2 فالأمر يختلف كما أسلفتُ القول.

و هذا النوع من المشاكل في الأكواد يُعد هو الأسوأ؛ لأنه لا يظهر في المراحل الأولي من كتابة البرامج، بل في مرحلة تنفيذها و الاعتماد عليها في العمل، و هي المرحلة التي لا يُمكن للمبرمج التدخل فيها لإصلاح الأمور إلا بخسائر كبيرة، و مع بذل مجهودٍ أكبر لفهم أسباب و ظروف و طرق حل المشكلة التي تحدث.

و حتي إن كانت هناك طبقةٌ قويةٌ من مُعالَجة الأغلاط exceptions handling فإن الأمور تظل تزداد سوءاً كلما ازداد حجم شجرة الوراثة، حتي نصل لمرحلةٍ نُضطر فيها إلي التضحية بما نعتقد وجوده من ميزات الوراثة الناقصة و إهمالها؛ حتي نستطيع الحصول علي أكوادٍ مأمونة.

- الزيادة المُطّرِدة في قواعد لغة البرمجة التي تدعم هذا النوع من الوراثة؛ فهي تحتاج علي الأقل لدعم القواعد التالية:
  - قاعدةٌ لتحديد أي المكونات الموروثة سيتم استثناؤها من الوراثة. فمثلاً في لغة إبداع (كما سبق التوضيح في المثال الوهمي السابق) يمكننا أن نستخدم قاعدة ما عدا. و التي تؤدي عملاً قريباً نوعاً ما في عملية الوراثة المتعددة (مع اختلاف الأسباب و النواتج). و مثل هذه القاعدة لا يمكن الاستغناء عن وجودها في أي لغةٍ تريد دعم الوراثة الناقصة.

  - قاعدةٌ لتحديد أي المكونات التي في الصنف القابل للوراثة نرغب في ألا يكون من الممكن استعمال الوراثة الجزئية معها؛ لأنه في معظم الأحيان سوف ترغب في إجبار المبرمج الذي سيستخدم الأصناف التي تكتبها علي عدم استثناء أي مكونٍ داخلها؛كنوعٍ من ضمان الاستخدام الأفضل لما كتبتَه من أكواد.

  و يمكننا عمل مثل هذه القاعدة بأكثر من طريق. فمثلاً يمكن استخدام الصفة التخيلية لازم في المثال التخيلي التالي:

```
صنف صنف 1 :
    إجراء لازم إجراء 1 :
        أكتب.نص.سطر("من داخل الإجراء إجراء 1 ")
```

حيث أن لازم هنا معناها أن الإجراء إجراء 1 من اللازم وجوده في الأصناف الوارثة للصنف صنف 1 . و بالتالي لو تمت كتابة الكود التالي:

```
صنف 1 كائن 1 = صنف 2()
كائن 1_إجراء 1 ()

صنف صنف 1 :
    إجراء لازم إجراء 1 :
        أكتب.نص.سطر("من داخل الإجراء إجراء 1 ")
```

صنف صنف 2 يرث صنف 1 :

ما عدا:

صنف 1 _إجراء 1

فإنه سيتم إعلان وجود خطأٍ فيه في مرحلة زمن التصحيح compile time؛ لأنه لا يجوز استثناء الإجراء إجراء 1 من الوراثة بينما تم إعطائه صفة لازم.

- عند تَضخُّم شجرة الوراثة، و كثرة عدد الأصناف الوارثة و تشابكها في سلسلةٍ طويلة: سوف نجد أن صعوبة فهم هذا النموذج و تتبع خصائصه تُعد أكبر منها في حالة الوراثة الكاملة؛
فالوراثة الكاملة رغم تسببها في تكبير حجم شجرة الوراثة إلا أنها بسيطةٌ في مفهومها الأساسي أن "كل صنفٍ وارثٍ يحوي كل المكونات التي في الصنف الموروث". بينما الوراثة الناقصة تحتاج منك للتدقيق في أي المكونات تم وراثته و أيها تم الاستغناء عنه، و في حين أن هذا يُعد أمراً بسيطاً في الوراثة بين عددٍ صغيرٍ من الأصناف فإنه حينما تكون شجرة الوراثة كبيرةً فإن الأمر يتحول لبحرٍ من الرمال المتحركة.

الخلاصة

كانت فكرة الوراثة الناقصة فكرةً قديمةً طرأت علي ذهني أثناء الأيام الأولي من تصميم لغة إبداع. و كنتُ قد أحببتُ تلك الفكرة و أردتُ الحصول علي مميزاتها في نموذج الوراثة المُعتمَد في اللغة. لكن إعادة التفكير في العيوب الشنيعة التي فيها جعلتني أُبعِدها عن الوجود في الشكل الحالي للقواعد القياسية في إبداع.

و أسأل الله تعالي أن أكون قد وُفِّقتُ في توضيح ما أردتُ توضيحه من نقاط.